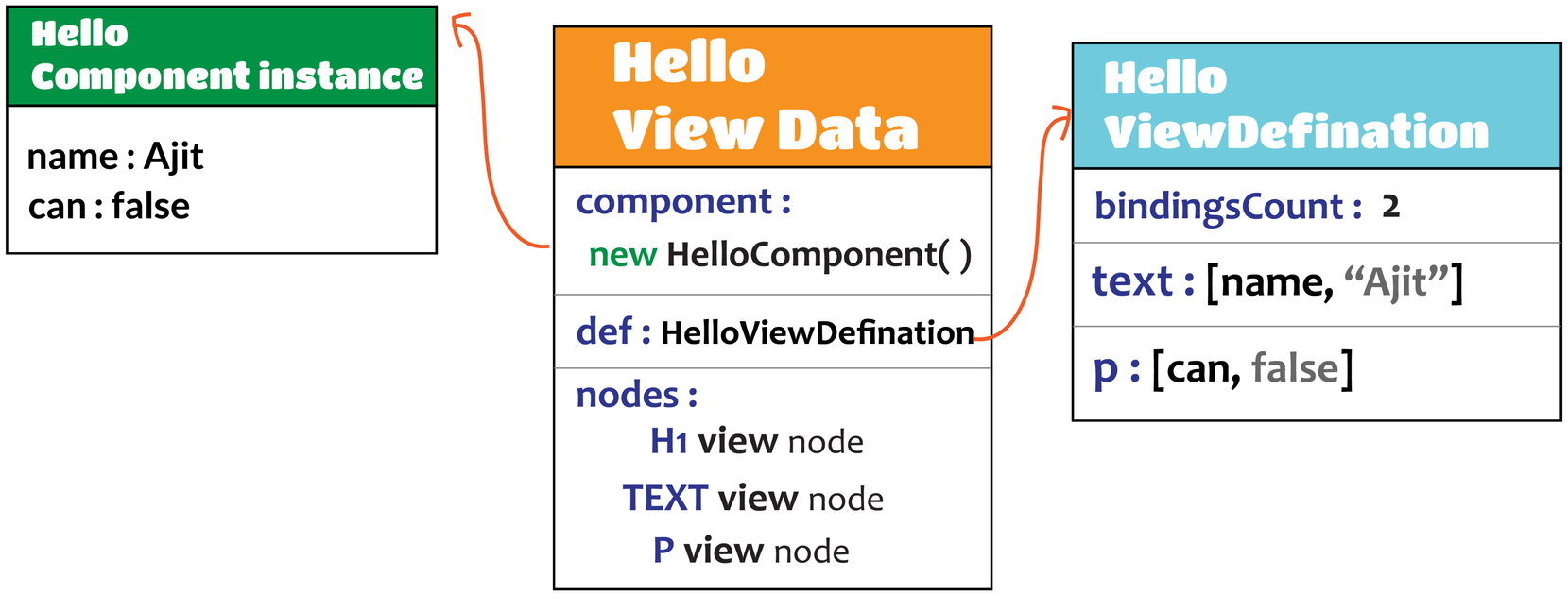
hello.component.html

```
<h1> Hello </h1>

{{ name }}

<p *ngIf="can"> Sir </p>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
   name = ' Ajit ' ;
   can =  false ;
}
```



internal conversion/representation

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="can"> Sir </p>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  name = 'Ajit';
  can = false;
}
```

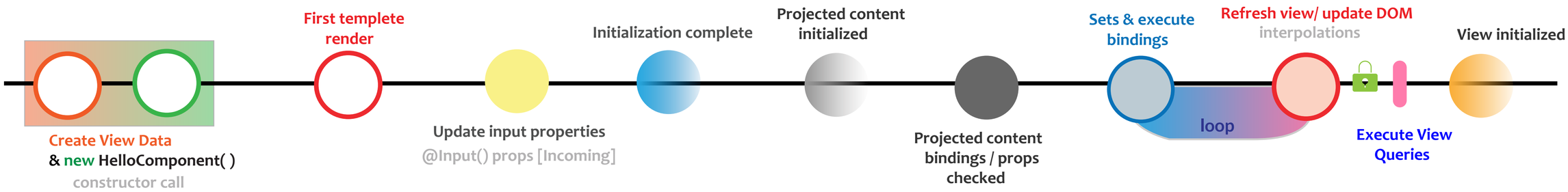
Hello View Data

- instance (component instance ref)
- bindingsCount : 2
- name (binding)
- can (binding)

Hello Component instance

name : Vinay ~~Ajit~~
can : false

(class instance)



Hello View State (LOCKED)

name: Ajit
can: false

Hello Execute / Verify Changes

name: Vinay ✗
can: false

MAIN STACK

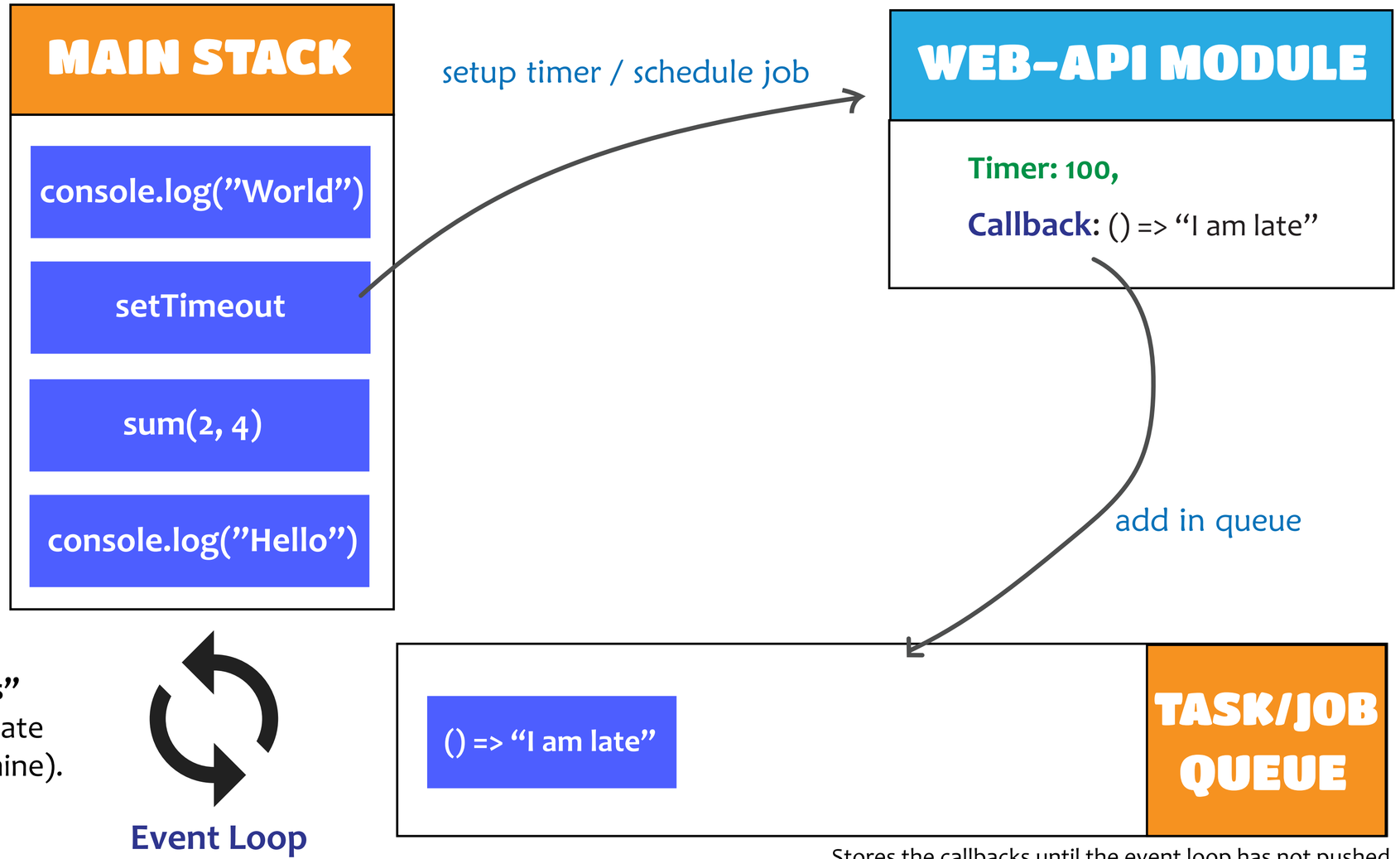
- Verification Loop
- Life cycle process completed
- this.name = Vinay
- View Initialized
- Life cycle started

Viewport

Hello

Vinay

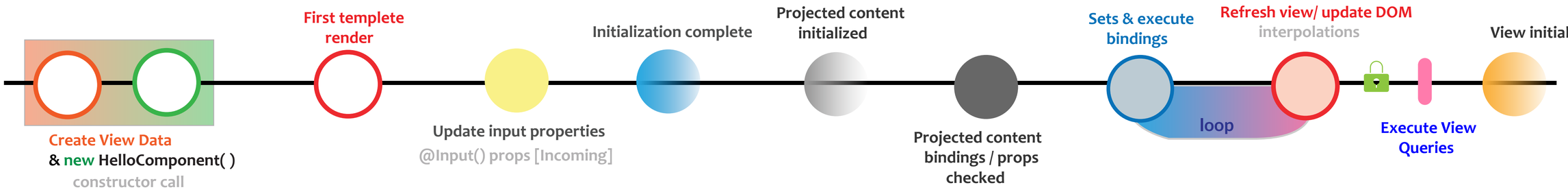
```
console.log(”Hello”);

sum(2, 4);

setTimeout(() => “I am late”, 100);

console.log(”World”);
```

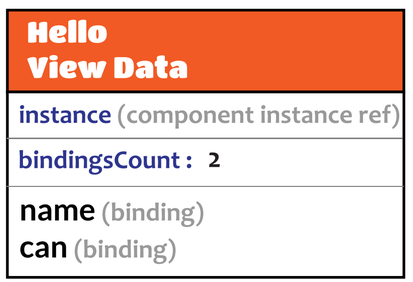


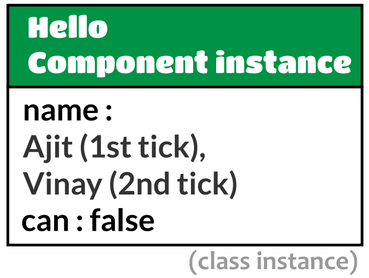
One complete cycle of **“continuous executions”** in the main stack from its **filled** state to **empty** state is called "One tick" of Javascript VM (Virtual machine).

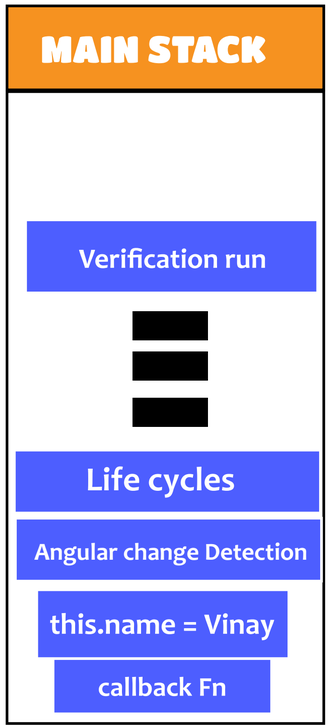
When stack is empty/free (1 tick completed)
Check queue
Place in stack/execute (2nd tick)

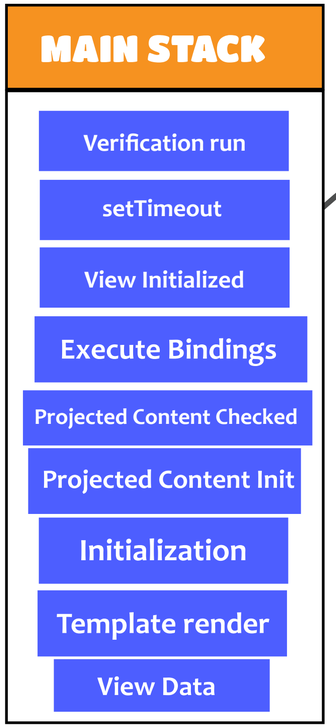
Stores the callbacks until the event loop has not pushed them into the main stack **(until the main stack is free)**

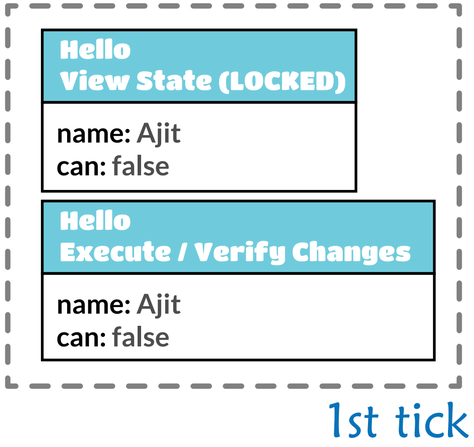

First templete render
Initialization complete
Projected content initialized
Sets & execute bindings
Refresh view/ update DOM
interpolations
View initialized
Create View Data
& new HelloComponent( )
constructor call
Update input properties
@Input() props [Incoming]
Projected content bindings / props checked
loop
Execute View Queries

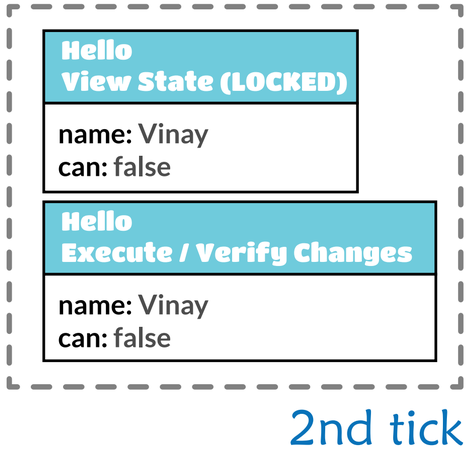

Hello View Data
instance (component instance ref)
bindingsCount : 2
name (binding)
can (binding)

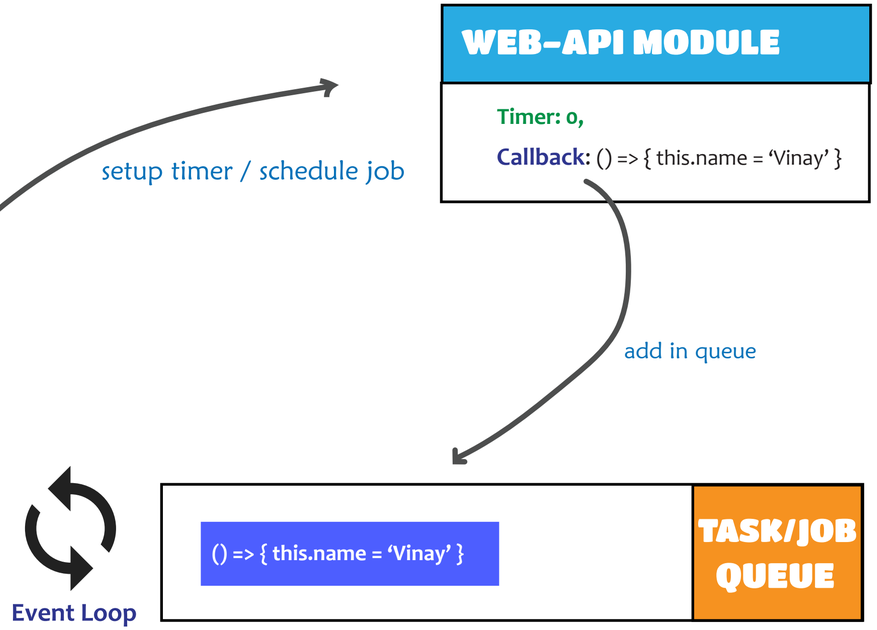

Hello Component instance
name :
Ajit (1st tick),
Vinay (2nd tick)
can : false
(class instance)


MAIN STACK
Verification run
Life cycles
Angular change Detection
this.name = Vinay
callback Fn
2nd tick


MAIN STACK
Verification run
setTimeout
View Initialized
Execute Bindings
Projected Content Checked
Projected Content Init
Initialization
Template render
View Data
1st tick


Hello View State (LOCKED)
name: Ajit
can: false
Hello Execute / Verify Changes
name: Ajit
can: false
1st tick


Hello View State (LOCKED)
name: Vinay
can: false
Hello Execute / Verify Changes
name: Vinay
can: false
2nd tick


setup timer / schedule job
WEB-API MODULE
Timer: 0,
Callback: () => { this.name = 'Vinay' }
add in queue
Event Loop
() => { this.name = 'Vinay' }
TASK/JOB QUEUE

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="can"> Sir </p>
<greet [message]="message"> </greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  name = ' Ajit ' ;
  can = false ;
  message = 'Morning';
}
```

Hello View Data

instance (component instance ref)

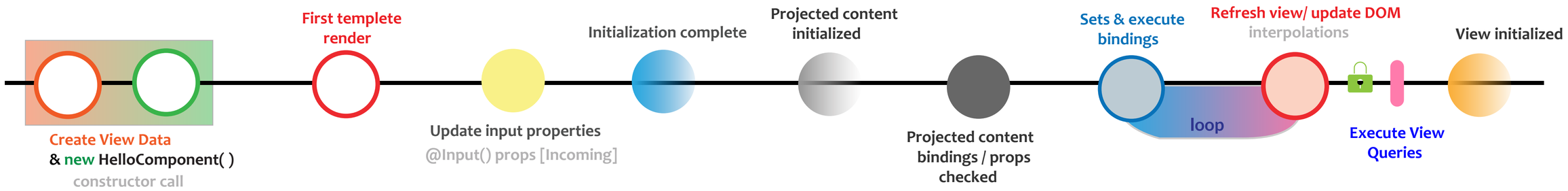
bindingsCount : 3

name (binding)
can (binding)
message (binding)

Hello Component instance

name : Ajit
can : false
message : Good Morning ~~Morning~~

(class instance)



Hello View State (LOCKED)

name: Ajit
can: false
message: Morning

Hello Execute / Verify Changes

name: Ajit
can: false
message: Good Morning ❌

MAIN STACK

Verification Loop
Life cycle process completed
this.message = 'Good Morning'
View Initialized
Life cycle started

Viewport

Hello

Ajit

Greet with Good Morning

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent)
  greetCmp: GreetComponent;
  name = ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)

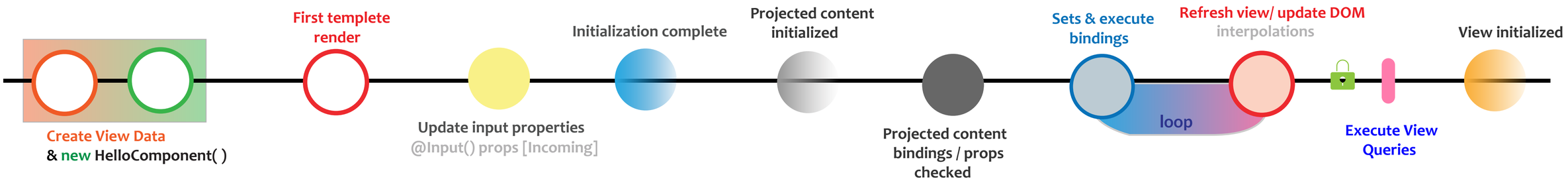
bindingsCount : 2

name (binding)
greetCmp (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp ~~undefined~~
name : Ajit

(class instance)



Hello View State (LOCKED)

name: Ajit
greetCmp: undefined

Hello Execute / Verify Changes

name: Ajit
greetCmp: GreetComp instance ✗

greetCmp instance gets defined

MAIN STACK

- Verification Loop
- Life cycle process completed
- Execute view queries
- View Initialized
- Life cycle started

Viewport

Hello

Ajit

For you

Flowers

hello.component.html

```
<h1> Hello </h1>
 {{ name }}
<p *ngIf="greetCmp"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
   @ViewChild(GreetComponent, { static: true })
    greetCmp: GreetComponent;
    name = ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)

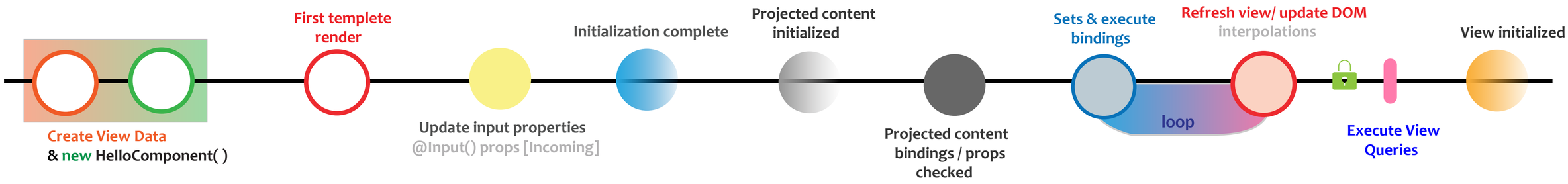
bindingsCount : 2

name (binding)
greetCmp (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)



Hello View State (LOCKED)

name: Ajit
greetCmp: GreetComp instance

Hello Execute / Verify Changes

name: Ajit
greetCmp: GreetComp instance

MAIN STACK

- Verification Loop
- Life cycle process completed
- Static resolve child queries
- Template render
- View Data
- Life cycle started

Viewport

Hello

Ajit

For you

Flowers

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp.isAllowed"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent)
   greetCmp: GreetComponent;
   name = ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)

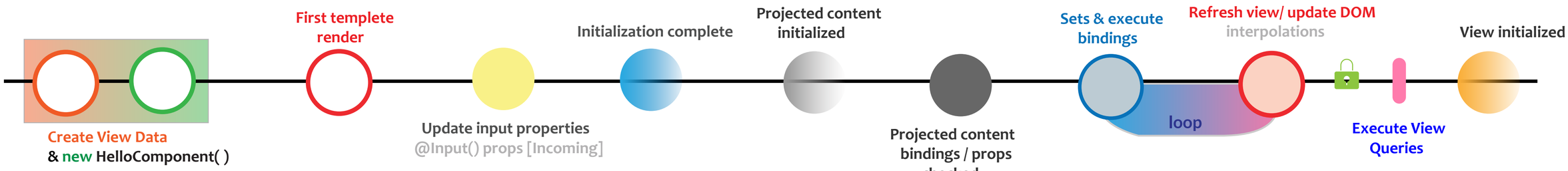
bindingsCount : 2

name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

Hello Component instance

greetCmp : GreetCmp ~~undefined~~
name : Ajit

(class instance)



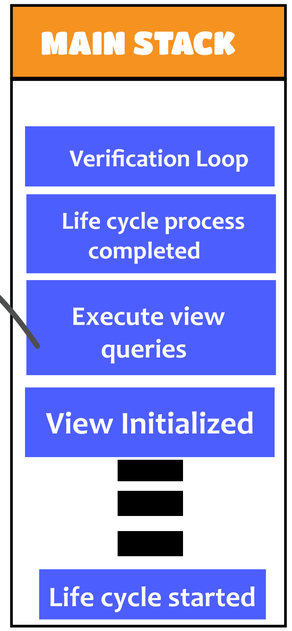
Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: null

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

greetCmp instance gets defined



Viewport

Hello

Ajit

For you

Flowers

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp.isAllowed"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
  greetCmp: GreetComponent;
  name = 'Ajit';
}
```

Hello View Data

instance (component instance ref)

bindingsCount : 2

name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

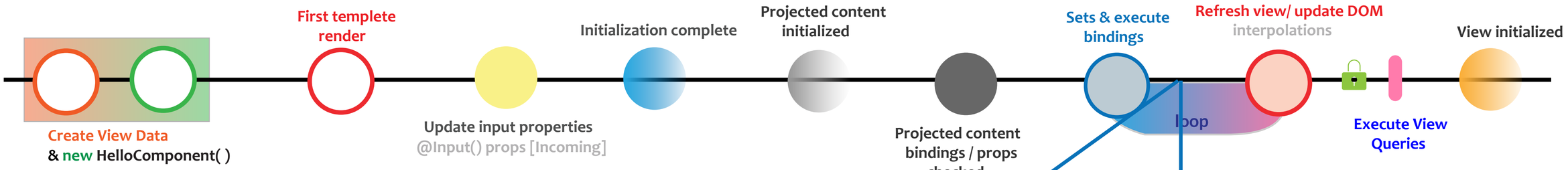
Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true ~~false~~ (ngOnInit)



Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: false

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

MAIN STACK

- Verification Loop
- Life cycle process completed
- app-greet initialization
- Static resolve child queries
- Life cycle started

Viewport

Hello

Ajit

For you

Flowers

According to Hello View Data
1. The name binding executed
2. The greetCmp.isAllowed binding executed
3. The app-greet initialization
4. The app-greet initialization causes isAllowed = true

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<app-greet> </app-greet>
<p *ngIf="greetCmp.isAllowed"> For you </p>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
   greetCmp: GreetComponent;
   name = ' Ajit ' ;
}
```

Hello View Data

instance (component instance ref)

bindingsCount : 2

name (binding)
app-greet (component)
greetCmp.isAllowed (binding)

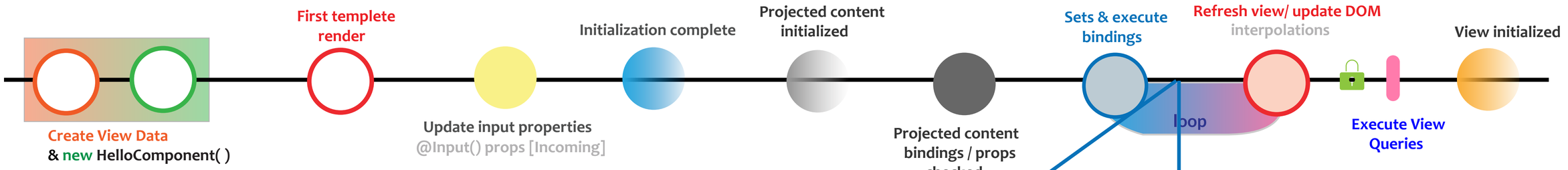
Hello Component instance

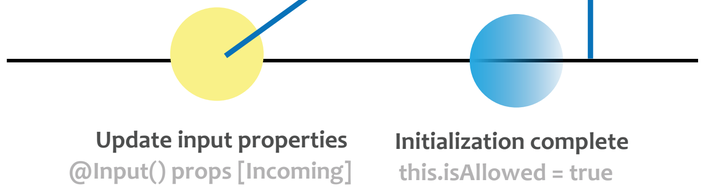
greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true ~~false~~ (ngOnInit)





Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: true

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

MAIN STACK

- Verification Loop
- Life cycle process completed
- app-greet initialization
- Static resolve child queries
- Life cycle started

Viewport

Hello

Ajit

Flowers

For you

According to Hello View Data
1. The name binding executed
2. The app-greet initialization
3. The app-greet initialization causes isAllowed = true
4. The greetCmp.isAllowed binding executed

hello.component.html

```
<h1> Hello </h1>
{{ name }}
<p *ngIf="greetCmp.isAllowed"> For you </p>
<app-greet> </app-greet>
```

hello.component.ts

```
@Component({ ... })
class HelloComponent {
  @ViewChild(GreetComponent, { static: true })
  greetCmp: GreetComponent;
  name = 'Ajit';
}
```

Hello View Data

instance (component instance ref)

bindingsCount : 2

name (binding)
greetCmp.isAllowed (binding)
app-greet (component)

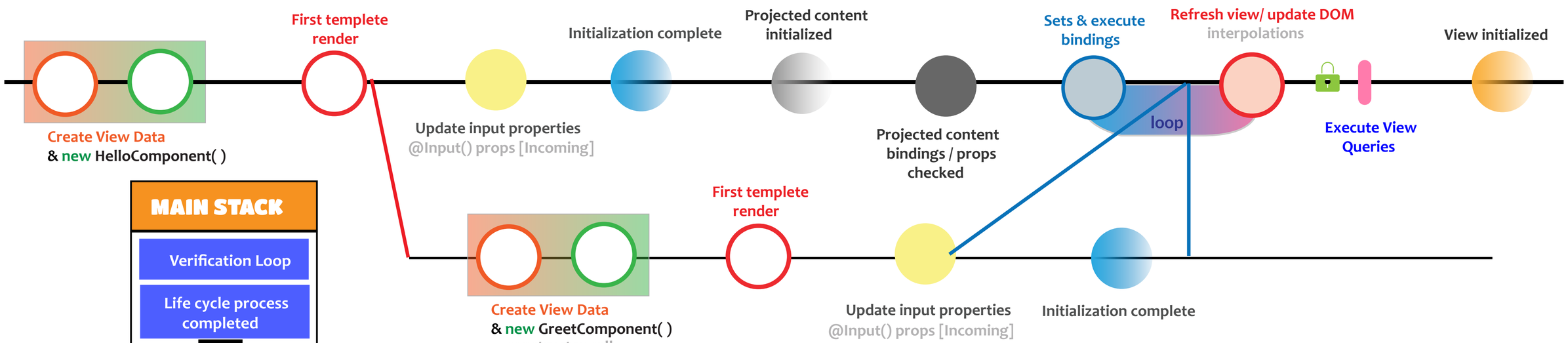
Hello Component instance

greetCmp : GreetCmp (static resolve)
name : Ajit

(class instance)

Greet Component instance

isAllowed: true (constructor)



MAIN STACK

- Verification Loop
- Life cycle process completed
- app-greet view Data
- Static resolve child queries
- Life cycle started

1. The app-hello view data
2. The app-greet view data, constrcutor call causes isAllowed = true
3. The name binding executed
4. The greetCmp.isAllowed binding executed

Hello View State (LOCKED)

name: Ajit
greetCmp.isAllowed: true

Hello Execute / Verify Changes

name: Ajit
greetCmp.isAllowed: true

Viewport

Hello

Ajit

For you

Flowers

Hello
View Data

instance (component instance ref)

bindingsCount : 1

isLoading (binding)

Hello
Component instance

isLoading : true (initially)

(class instance)

Hello
View State (LOCKED)

isLoading: true

Hello
Execute / Verify Changes

isLoading: false

startWith case

Hello
View State (LOCKED)

isLoading: true

Hello
Execute / Verify Changes

isLoading: true

without startWith case

1. Call operators and store values (if any)
2. [startWith: V, tap: **Fn**]
3. If any operator has return a value, store it
4. Call Subscribe and **register** callabck Fn
5. If any value V already present
   a. Call pipe operators **registred** callbacks [tap: **Fn**]
   b. Call **registered** callback Fn (subscriber)

   else
   Wait for value (Next tick), then repeat step 5

VALUE: [ ]

Pass to below FN(s)
(if value is present)

() =>
this.isLoading = false

() =>
console.log
("posts", posts)

RxJs Registered callbacks

MAIN STACK

Verification run

call subscrbe registred Callback

this.isLoading = false

call -> tap: registred Callback

call pipe operators callback

subscribe: Register Callback

tap: Register Callback

startWith: Return value

pipe

getJSON

View Initialized

Execute Bindings

Life cycle started

setup ajax /
schedule job

WEB-API MODULE

Ajax: Http Request

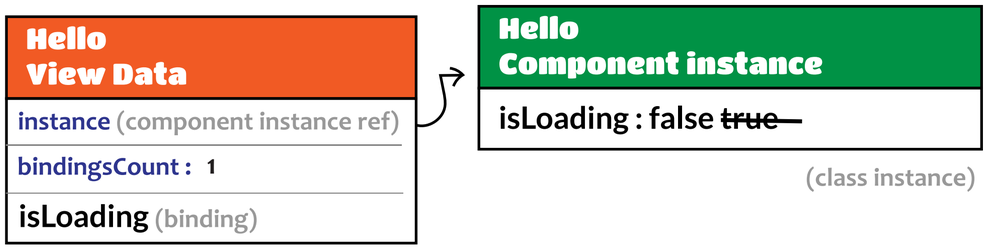
Callback: () => RxJs Observable

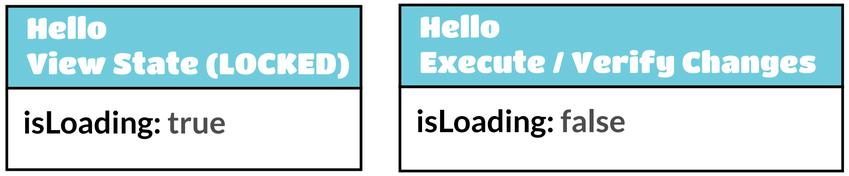
add in queue

Event Loop

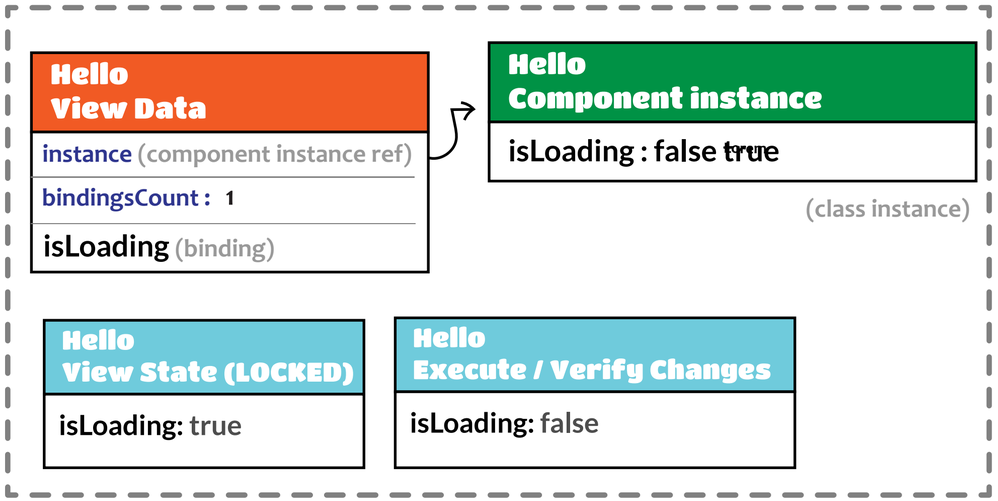
RxJs Observable
http request data

TASK/JOB QUEUE

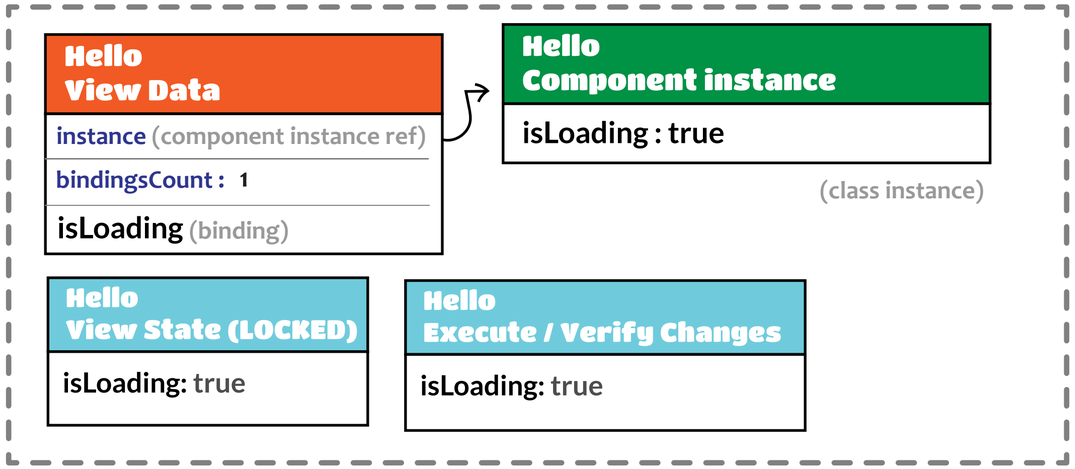




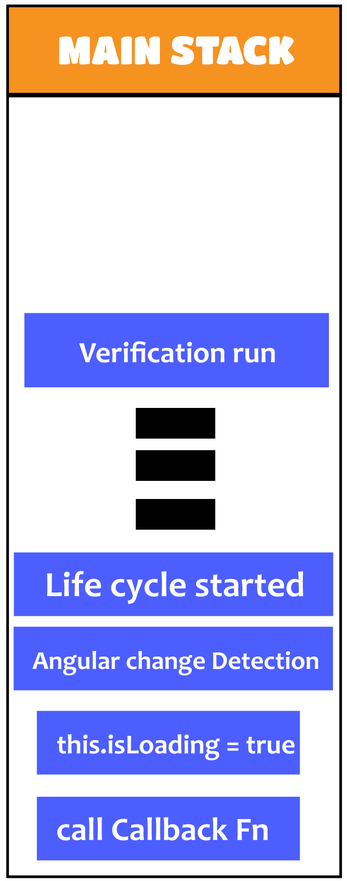
1. Call **Promise constructor** with callback Fn
2. Call **callback** Fn
3. If **resolve** is called, call the **then** registred callback
4. If **reject** is called, call the **catch** registered callabck



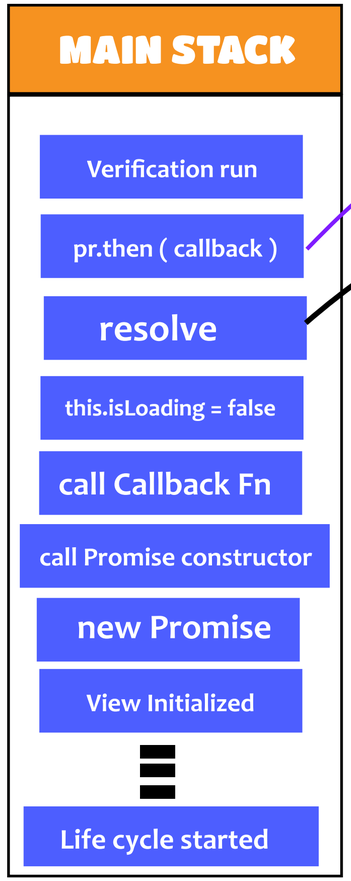
1st tick



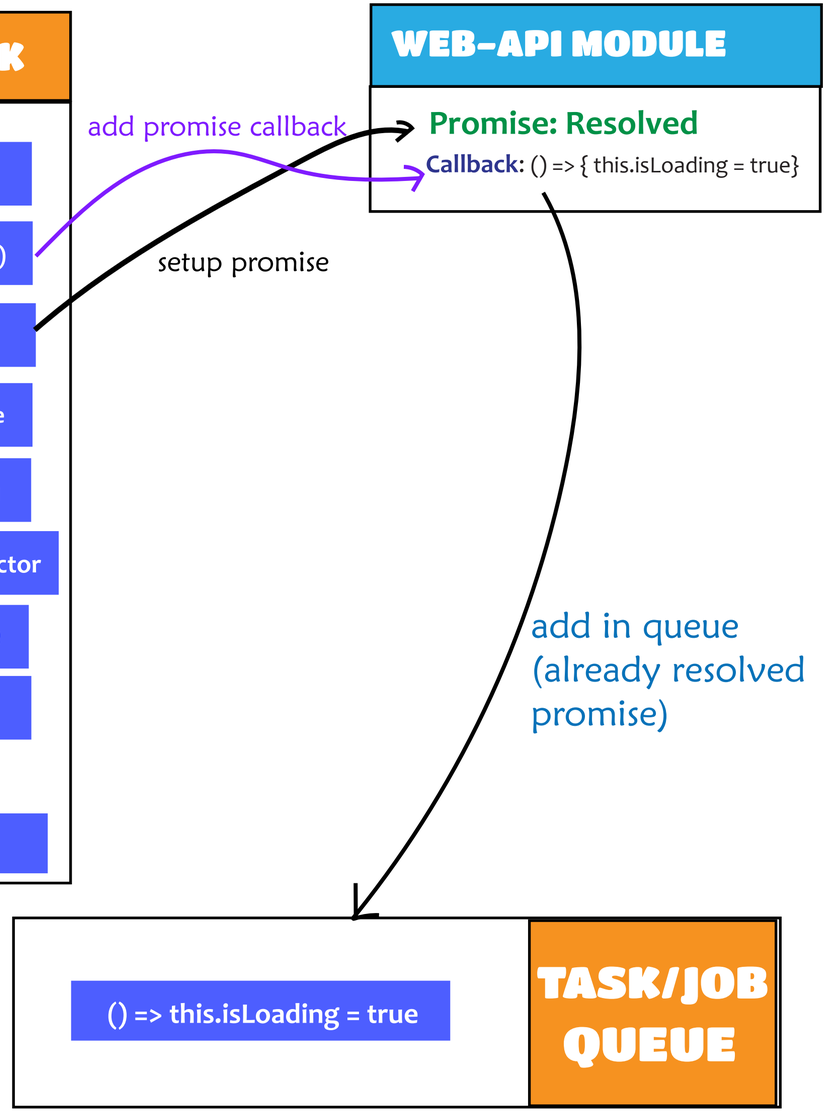
2nd tick



2nd tick



1st tick



1. Call **Promise constructor** with callback Fn
2. Call **callback** Fn
3. If **resolve** is called, call the **then** registred callback
4. If **reject** is called, call the **catch** registered callabck